(2)

ONR

RELATIONAL PROCESSOR EXPERIMENTS (FINAL REPORT)

June 21, 1991

Stephanie Lockhart
Kevin Walker
Pat Watson

IBM Corporation 250/060
9500 Godwin Dr
Manassas, VA 22110

N00014-88-C-0745

Contents

1.0 Introduction

For the past several years IBM, the Office of Naval Research (ONR), and the Distributed Command and Control (DC2) project at the Naval Ocean System Center (NOSC), San Diego have co-sponsored complementary distributed real-time system research at Carnegie Mellon University (CMU) and the University of Virginia (UVA). One area of IBM concentration that has become of increasing importance to ONR and the DC2 project is real-time distributed database management. IBM, because of its involvement in the design of real-time submarine combat systems, had to address the real-time issues in this area earlier than most system architecture researchers. Real-time systems which need very fast as well as predictable response time performance cannot use standard commercial approaches to database resource and consistency management. In order to provide acceptable response for critical transactions, real-time database managers should use response driven transaction scheduling, e.g., rate-monotonic techniques. This implies that database managers should use a precedence mechanism (such as priority) that is based on transaction response requirements for selecting which transaction to process next and they should support the notion of preemption so that transactions with higher precedence may move ahead of already executing lower precedence transactions. To avoid unbounded precedence (or priority) inversions resulting from database consistency management requirements, techniques such as the priority ceiling protocol (ref 1) which avoids deadlock, minimizes blocking, and makes response times more predictable should be used. To provide fast response, memory-based database managers are needed as well as special hardware designed for highly efficient processing of relational queries.

IBM has been very involved in designing systems where these characteristics are essential. IBM reached the prototyping stage for real-time relational database technology in 1990, i.e., the technology became available for experimenting with different application domains such as command and control, surveillance systems, etc. Under IBM and now NOSC/ONR sponsorship, CMU and UVA are implementing a real-time relational database manager which will support the scheduling concepts already developed by CMU and UVA. CMU and UVA are integrating the real-time relational database manager into a real-time operating system in order to prove the validity of the scheduling theory.

IBM acquired an Ada programmable Versa Module Europa (VME) based relational processor (RP) which is capable of at least one order of magnitude more transactions per second than is typical of currently available relational database software packages. While this special relational database hardware is not preemptable, transactions can be scheduled by its control processor. Because of its speed it can support real-time systems where use of a software/disk based relational database manager would not provide acceptable response performance. Furthermore, given the semantic knowledge derived from a particular application's implementation it is possible to bound transaction execution times so that closed form scheduling analysis and prediction techniques can be supported.

The remainder of this paper describes the experiments that IBM has performed for NOSC/ONR in an initial evaluation of the relational processor as applied to Navy command and control trackfile management problems. This effort uses the same track data as NOSC's Distributed Operating System Experiments (DOSE) research task (ref 2).

1.1 Relational Processor Experiment Description

1.1.1 Hardware Configuration

The Real-Time Relational Processor testbed consists of the Ferranti International Relational Processor (ref 3) and a DY-4 single board computer (SBC) (ref 4) configured on a VME chassis. The Relational Processor (RP) contains hardware that executes relational primitives faster than traditional database software. The standard 8 megabyte Relational Processor is a 2 card VME module to which additional memory cards can be added to bring the total available RP memory to 104 megabytes. The VSB (VME sub bus) provides an interface between the Relational Processor and the 68020-based DY-4 DVME-137 SBC. A Sun workstation connected to the SBC by 2 serial ports currently serves as a monitor, which passes transaction requests and data to and from the SBC that controls the relational processor. This is illustrated by Figure 1.

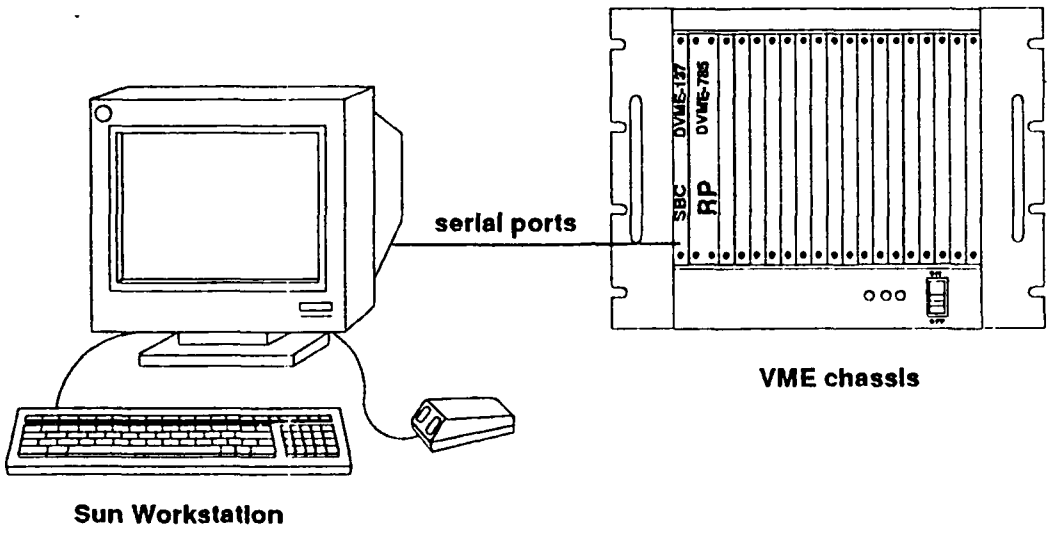

Figure 1. Hardware Configuration

1.1.2 Software Configuration

The software is primarily composed of Ada control structures and relational primitive bindings (SQL is not used). Ferranti provides the Ada bindings for the relational primitives in a Program Interface Library (PIL). The relational primitives are similar to the SQL query language. The Ada application software is compiled on a Telesoft compiler and executed with Ready Systems ARTX run time environment.

1.1.3 DOSE Experiment

Binary data enters DOSE in the Network Interface Unit (NIU) Access Machine and is passed to the parser manager. The parsed messages are then stored on the database of the Track Report Manager and displayed by the Graphics Map Client. This is depicted in Figure 2 on page 3.

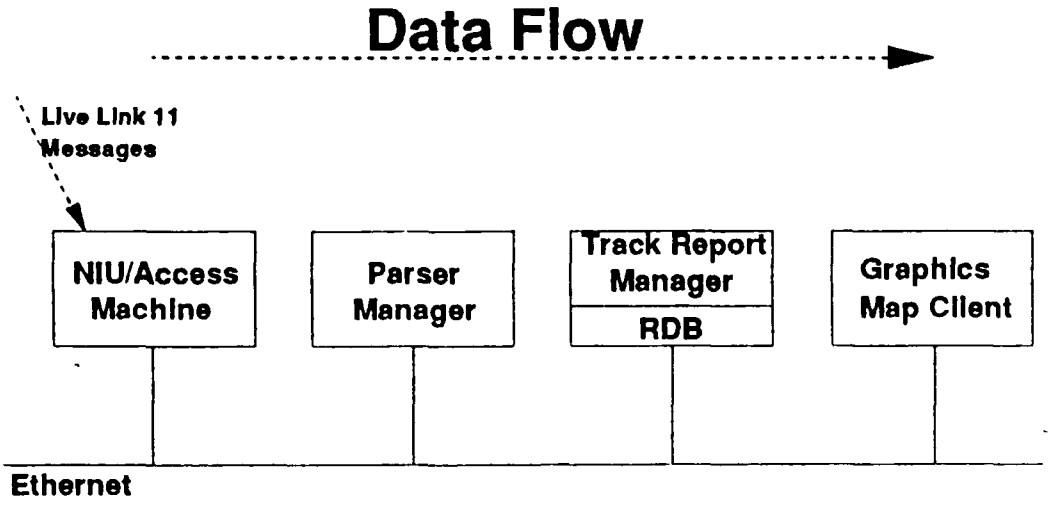

Figure 2. DOSE Data Flow

The DOSE track data describes a set of contacts over time providing track number, position, movement, and classification information. The track data has 17 attributes consisting of floating point, characters, and integers.

The data enters the system in blocks of 2 to 57 tracks which represent an update to the scenario. The database is searched for each track number in the current block. If a match is found, the entry for that track number is updated and given the new time stamp. If the track number is not found in the database, a new track is inserted.

In addition to maintaining history on the contacts, the DOSE graphics processor may request data by track number or time stamp for display updates. Either query will return all records matching the selection criteria.

The DOSE database operations are relatively simple and are largely characterized by the following four queries. The queries used in the experiment are given here in their original SQL format and in the equivalent RP primitives.

1. INSERTION - Add a track for the new track number.

   SQL:
   INSERT INTO relation VALUES (track, latitude, ..., nuclear)
   Relational Primitives:
   DEFINE_TUPLES_TO_ADD(buffptr, relation, tuple, words)
   DEFINE_ATT_VAL(buffptr, number)
   DEFINE_ATT_VAL(buffptr, value)
   DEFINE_ATT_VAL(buffptr, value)
   ...
   DEFINE_ATT_VAL(buffptr, value)

   Note: "buffptr" indicates the command/input/output buffer being used.

2. UPDATE - Update previous track with new data.

   SQL:
   UPDATE relation SET attribute = value WHERE track = number
   repeated for each attribute.
   Relational Primitives:
   SELECTION(buffptr, EQ, tla, track, number)
   UPDATE_ATT(buffptr, track, tla, number)
   UPDATE_ATT(buffptr, latitude, tla, value)
   UPDATE_ATT(buffptr, longitude, tla, value)
   ...
   UPDATE_ATT(buffptr, nuclear, tla, value)

   Note:"tla" is the pointer set that relates the SELECTION results to the UPDATE_ATT request.

3. TRACK SELECTION - If found, return the track with the requested track number.

   SQL:
   SELECT * FROM relation WHERE track = number
   Relational Primitives:
   SELECTION(buffptr, EQ, tla, track, number)
   OUTPUT(buffptr, tla, 0, 0)

4. TIME SELECTION - If found, return all tracks with the requested time.

   SQL:
   SELECT * FROM relation WHERE time = gmt
   Relational Primitives:
   SELECTION(buffptr, EQ, tla, time, gmt)
   OUTPUT(buffptr, tla, 0, 0)

1.2 Time Measurement Methodology

To measure the required processing time for a RP transaction, a 10 ms system clock was available. Because individual database transactions execute on the RP in much less time than a single 10 ms clock tick, it was necessary to perform multiple iterations of the same transaction to get an accurate measurement of the processing time. As the number of iterations increased to approximately 100, the actual processing time became apparent. Figure 3 on page 5 illustrates this point.

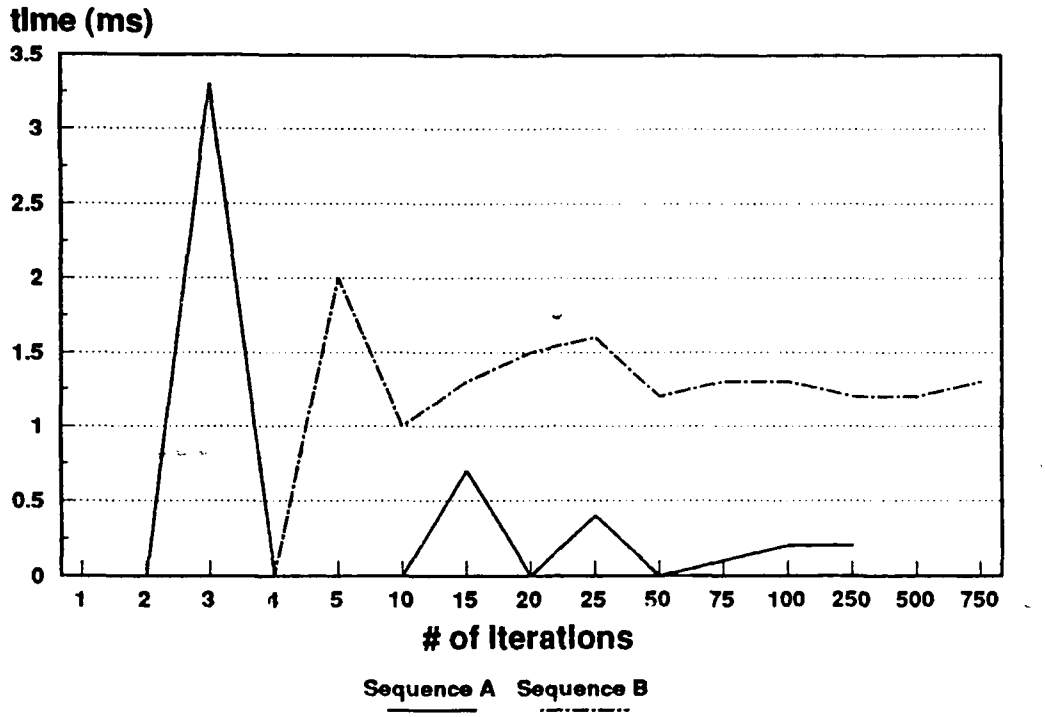

Figure 3. Update Attribute (1000 tuples)

Transaction processing time includes RP processing time, the time associated with transferring the command and input buffers across the VSB, as well as the overhead involved in the Ada constructs needed to duplicate the flow of control of the DOSE scenario. By synchronizing a timer with the signals sent across the VSB, it would be possible to eliminate all processing not occurring on the RP, but the method chosen provides a better estimate of the amount of time required for real-world applications.

The RP can be programmed to amortize transaction overhead processing over a few or many commands. Consequently, performance was measured for both approaches as described below.

1.2.1 Multiple Commands - Sequence A

In order to minimize communication over the VSB, the RP allows multiple commands to be placed in sequence in the command buffer. Certain operations may not appear more than once in these command buffers, but the basic functions required by our experiments are not of this nature. By grouping a number of commands and transmitting the buffer, all overhead except for a single VSB transaction was avoided, and the times resulting from this method are much lower than Sequence B where command blocking was not performed.

1.2.2 Multiple Iterations - Sequence B

The second method was to execute single transactions inside a loop, and divide the elapsed time by the number of iterations before converting to seconds. This method includes the amount of time necessary for loop control processing, but by placing clock queries within the loop and adding up the individual elapsed times, this overhead was found to be negligible. However, overhead associated with command and data transfers proved to be very significant.

1.3 *Insert Measurements*

An insert was performed on a database with 75 records (the approximate size of the DOSE database) and on a database of 1000 records. By increasing the number of iterations on the database with 75 records, the time required stabilized to 0.4 ms for Sequence A and 1.5 ms for Sequence B as illustrated in Figure 4. Increasing the size of the database to 1000 records has little impact on the response time as illustrated in Figure 5 on page 7.

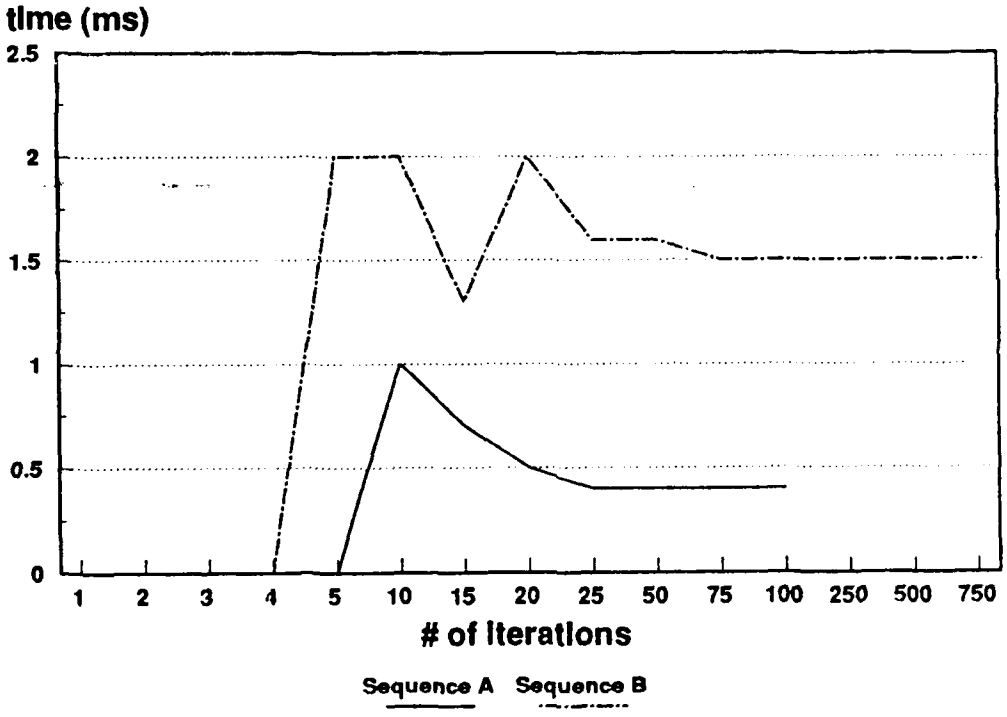

Figure 4. Insertion (75 tuples)

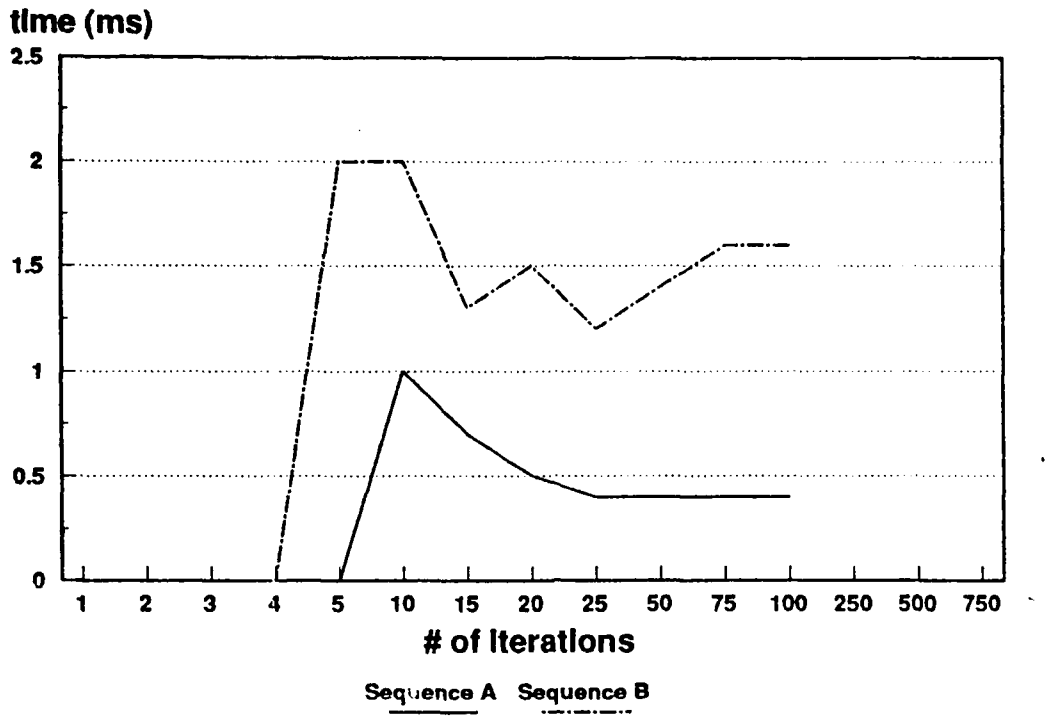

Figure 5. Insertion (1000 tuples)

1.4 Update Measurements

Update measurements were performed for updating an entire tuple which consists of 17 attributes as well as for a single attribute in the tuple.

Tuple update performance measurements were made on databases with 75 records and a 1000 records. In the database with 75 records, the resulting time stabilizes to 1.6 ms for Sequence A and 2.2 ms for Sequence B. Increasing the size of the table to 1000 records had an insignificant impact on the required processing time.

Next, tuple update was performed for various database sizes with a constant number of iterations. The RP is relatively insensitive to the number of tuples in the database as shown in Figure 6 on page 8.

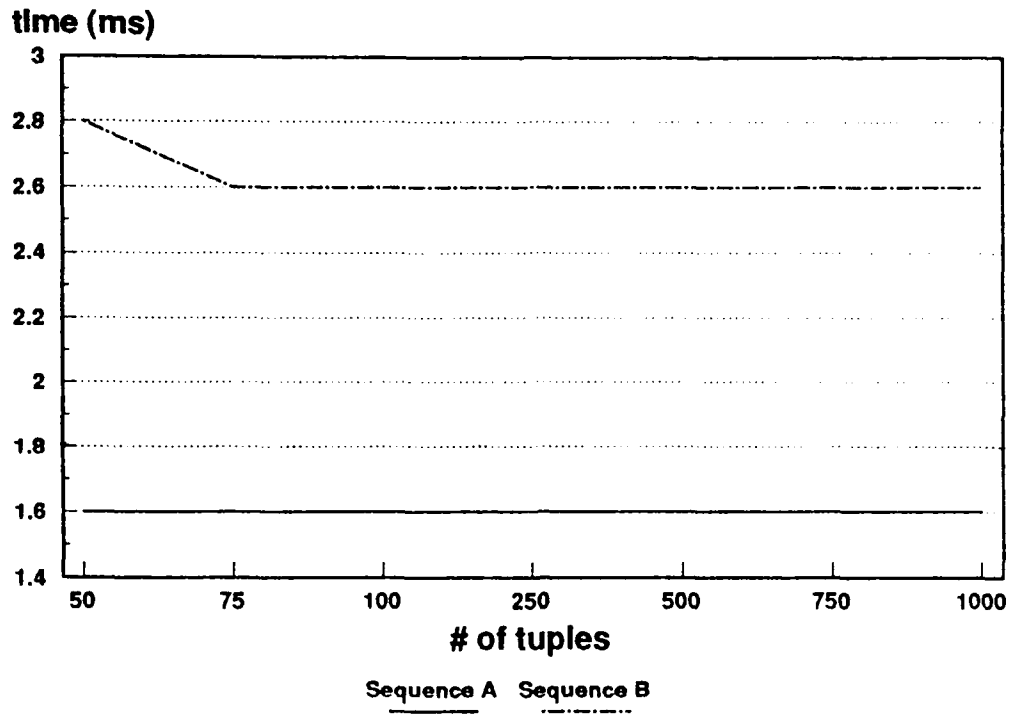

Figure 6. Update Tuple (50 iterations)

An attribute update was performed varying the number of iterations while maintaining the size of the database. For a database of 75 tuples, the resulting time stabilizes to 0.2 ms for Sequence A and 1.3 ms for Sequence B. The same results are seen for a database with 1000 tuples.

Also an attribute update was performed for various database sizes. Again, increasing the number of tuples had very little impact on the time required to perform the update as shown in Figure 7 on page 9.

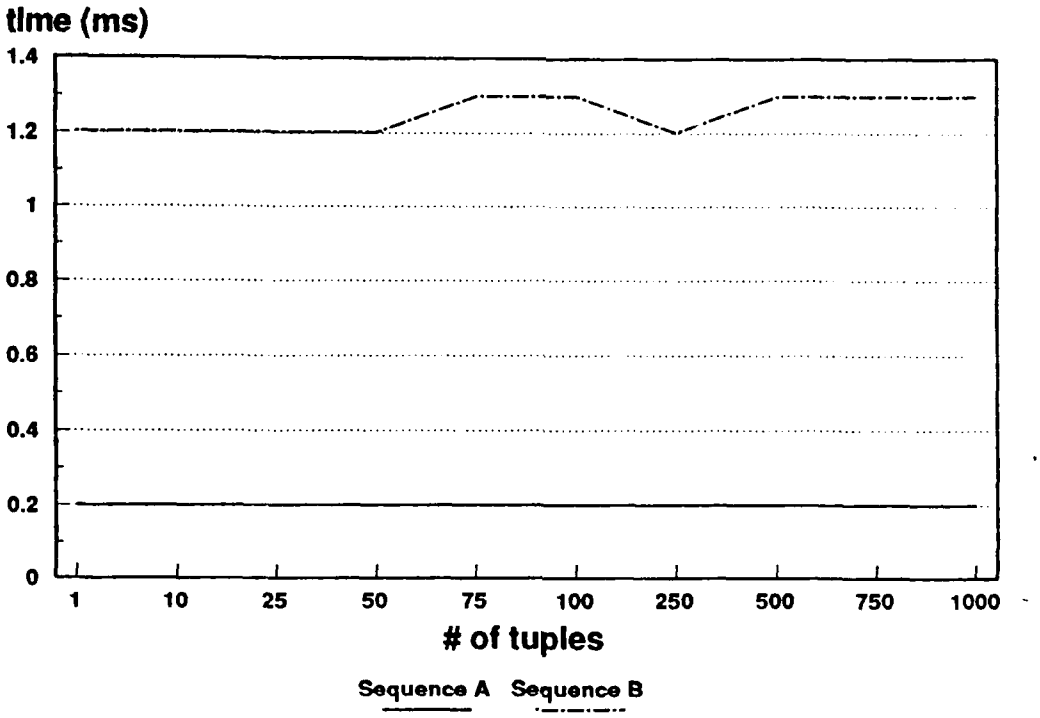

Figure 7. Update Attribute (250 iterations)

1.5 Selection Measurements

Track and time selections were measured in several ways.

A selection resulting in one matching tuple was measured over numerous iterations while maintaining the size of the database. For a "time" selection, it was found that the Sequence A time was 0.1 ms and for Sequence B the time was 3.7 ms. The "track" selection performs the select using the primary key and the resulting response time was the same as the above "time" selection.

Next, several selections were performed varying the number of matching tuples while maintaining the size of the database and the number of iterations. These results are shown in Figure 8 on page 10.

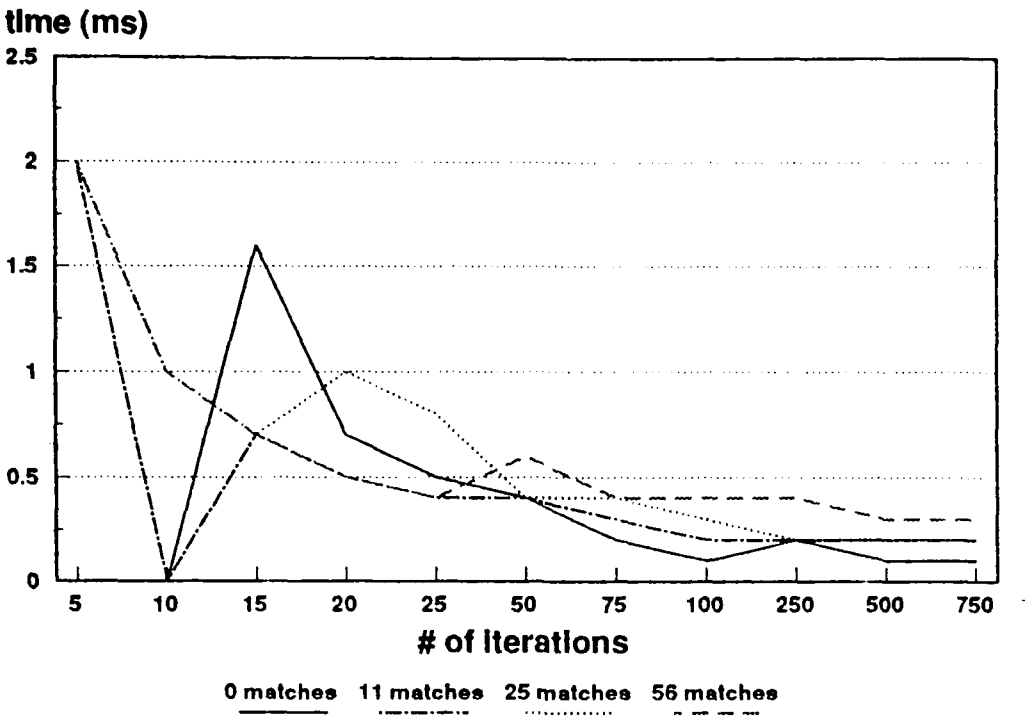

Figure 8. Selection (Sequence A, 75 tuples)

Finally, a selection was performed varying the size of the database while maintaining a constant number of iterations. Again the RP response time remained flat as the size of the database increased. Figure 9 on page 11 depicts these measurements.

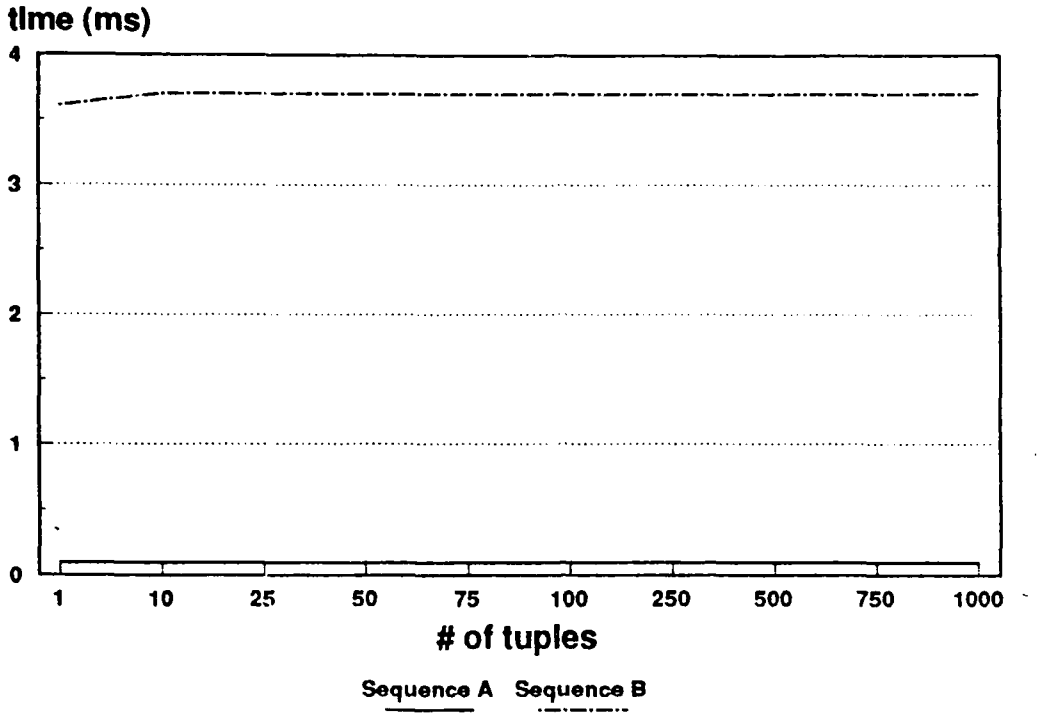

Figure 9. Selection (750 iterations)

1.6 Summary

Our experiments with the Relational Processor have shown it to be a viable technology for use in Navy command and control systems. In particular, the DOSE scenario requires that approximately 45 tracks per second be processed. Based on the experiments with Ferranti's Relational Processor it should be able to handle a sustained rate of 590 to 2375 tracks per second depending on the ratio of updates to inserts as determined by the existing database. This performance is more than an order of magnitude greater than required; however, these numbers would have to be derated by the amount of query activity directed at the Relational Processor, for instance, by the DOSE graphics processor. See Figure 10 on page 12 to obtain performance estimates for various mixes of track updates to inserts. While time did not permit measurements varying the number of attributes in a tuple, information obtained from the manufacturer indicates that the Relational Processor's performance appears to be more sensitive to the number of attributes than the number of tuples. All measurements in this experiment were made with the 17 DOSE attributes.

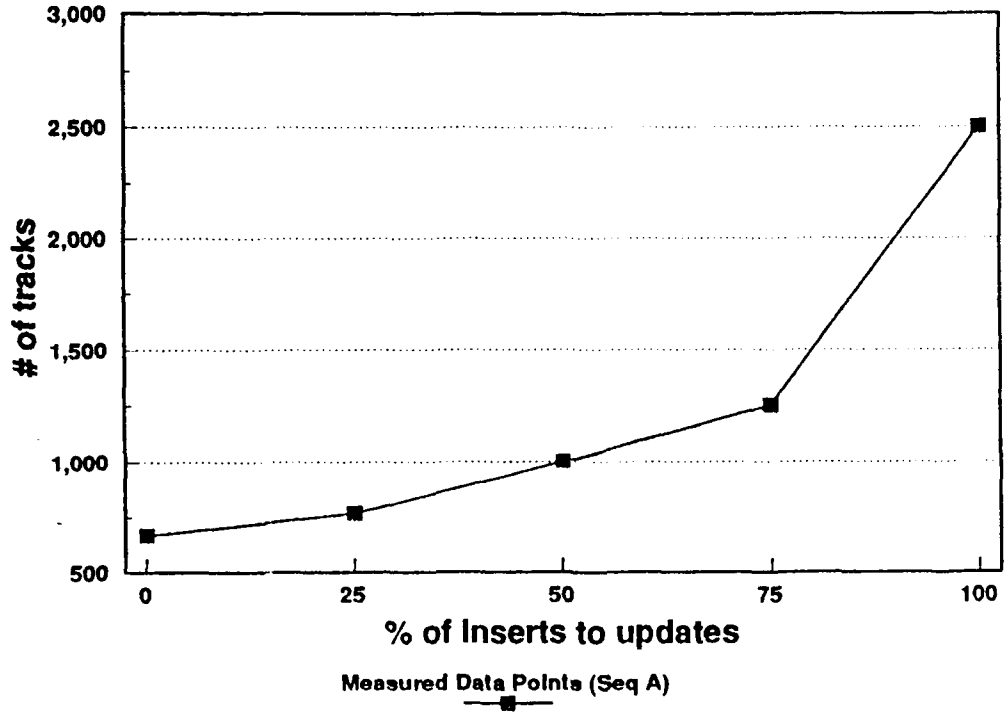

Figure 10. Mix of Inserts and Updates based on Selection

The experiments involved database sizes ranging from 75 to 1000 tuples. The Relational Processor's response characteristics proved to be flat over this range. The maximum size of the database which can be processed by the Relational Processor is restricted by the size of its volatile memory. The Relational Processor experiments used 8 megabytes but can be expanded to 104 megabytes.

Additional areas deserving future evaluation include time driven scheduling of the RP, performance sensitivity to databases larger than 1000 tuples and performance sensitivity to the number of attributes in a relation. Furthermore, it would be advantageous to exercise the Relational Processor in a distributed system network, permitting the measurement of additional forms of system overhead that could further degrade the deliverable performance of the Relational Processor in a real-world application.

This technology has great promise for future command and control as well as other applications where speed and insensitivity to database size are important characteristics. The inability to preempt the relational processor once it has started processing a command buffer has an impact on system transaction schedulability. However, given its performance, insensitivity to database extent, and semantic information about its use in particular applications, this weakness could often be overcome by judicious application programming such that bounded response time guarantees could be supported.

1.7 References

1. Sha, Lui, Ragunathan Rajkumar, and John P. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," IEEE Transactions on Computers, Vol 39, No 9, Sept 1990.

2. Butterbrodt, M.C. and J.C.M. Green, "The Distributed Operating System Experiment: A Vital Link in the Development of a Real-Time Relational Database Environment," Naval Ocean Systems Center, January 1990.

3. Ferranti Computer Systems Limited, "DVME-785 Relational Processor Hardware User Manual," Draft Issue - July 1988.

4. DY-4 Systems Inc., "Relational Processor Tutorial," February 1990.